VIOLA : 객체지향 데이타베이스를 위한 시각 질의 언어

(VIOLA : A Visual Query Language for Object-Oriented Database)

요약

본 논문에서는 ODMG 데이타 모델을 지원하는 객체지향 데이타베이스를 위한 시각 질의 언어인 VIOLA(VIsual Object-oriented query LAnguage)의 설계 및 구현에 대하여 제시한다. VIOLA는 기존의 범용적인 질의 언어(e.g.,SQL)와 유사한 명세 원칙에 기반하여 시각적으로 질의를 명세할 수 있도록 지원하며, 작성된 시각 질의는 OQL 질의 문장으로 변환되어 질의 처리기를 통해 실행된다. VIOLA는 기존의 관계형 질의 요소들(조건절, 프로젝션,한정자,집단화,그룹화) 뿐만 아니라 객체 지향 질의의 주요 요소들(컬렉션, 복합 객체, 경로 표현, 메소드)을 위한 시각적 명세 방법을 제공한다. 이를 위해 VIOLA는 중첩된 윈도우 형태의 시각 구조를 제공하며, 이 시각 구조는 모노이드 내포(monoid comprehension)에 기반한 세만틱을 가진다.[1]

Abstract

In this paper, we present the design and implementation of visual query language called VIOLA(VIsual Object-oriented query LAnguage) for object-oriented database supporting ODMG data model. With VIOLA, the user can specify query with visual manner based on the principle which is similar to that of popular query language such as SQL. VIOLA translates visual query into OQL query statement in order to be executed through the OQL query processor. It also supports visual specification method not only for components of relational query such as conditional predicate, join, quantifier, aggregate, and grouping, but also for major components of object-oriented query such as multiple collection, composite object, path expression and method. For this, VIOLA provides a visual construct based on nested window facility, and the construct has the semantics based on monoid comprehension.

[1] 본 연구는 통상 산업부의 공업 기반 기술 과제 943-20-4, "객체지향 데이타베이스를 위한 설계 도구의 구현"의 지원을 받아 작성된 것입니다.

1 서론

최근 객체지향 데이타베이스가 많은 각광을 받고 있다. 객체지향 데이타베이스는 풍부한 세만틱(semantics), 예를 들면, 상속성(inheritance), 참조 관련성(reference relationship) 및 집합 속성(set attribute)을 표현할 수 있는 데이타 모델을 제공하여, 멀티미디어, CAD(Computer-Aided Design), GIS(Geographic Information System) 등의 응용 분야에 쉽게 적용될 수 있다[1]. 하지만, 객체지향 데이타 모델은 다양한 세만틱을 제공하는 대신 구성이 복잡하게 되어, 이해하고 사용하기에는 어려워지게 된다. 또, 데이타베이스에 접근하기 위해 사용되어지는 데이타베이스 언어들은 비숙련자들이 사용하기에는 너무 어려워서 - 예를 들면, 언어 구문을 외워야 하고, 스키마 구조를 암기해야 하며, 복잡한 관련성 구조를 이해해야 한다 - 사용하는데 있어 많은 실수와 에러를 유발하고, 따라서, 데이타베이스 사용을 억제하는 요인이 되기도 한다[16]. 사용자가 데이타베이스를 쉽게 사용할 수 있도록 하기 위해서는 사용 전에 데이타베이스의 구조에 대한 충분한 정보가 제공되어야 하며, 비교적 단순하고 간단한 원칙에 기반한 질의 작성 방법이 제공되어, 그들의 직관과 감각을 충분히 활용할 수 있는 환경이 제공되어야만 한다[17].

본 연구에서는 객체 지향 데이타베이스에 대한 쉽고 간편한 접근 방법을 제공하기 위해 다음과 같은 특징들을 만족하는 시각 질의 언어를 개발하는 것을 목표로 하였다 :

1. 사용자는 몇 가지의 간단한 원칙만을 따라서 질의를 작성할 수 있어야 한다. 또, 이 원칙은 기존의 사용자들이 익숙하게 사용하였던 질의 작성 원칙과 크게 다르지 않아야 한다.

2. 몇 가지의 간단한 시각 요소만을 가지고 시각 질의를 작성할 수 있어야 한다.

3. 많이 사용되어지는 실제적인 객체 지향 질의 언어로 변환, 생성될 수 있어야 한다.

4. 정형적인 세만틱과 이론적인 배경을 가져야 한다.

본 논문에서는 위와 같은 목표를 이루기위해 고안된 시각 질의 언어인 VIOLA(VIsual Object-oriented query LAnguage)의 설계 및 구현 방법을 제시한다. 먼저, 첫번째 특징의 지원을 위해 VIOLA에서는 기존에 가장 널리 사용되고 있는 질의 언어인 SQL[10]이 취하고 있는 질의 명세 방법과 유사한 방법을 취하기로 하였다. SQL의 질의 명세 방법은 1) 질의의 대상이 되는 테이블을 선정하고, 2) 테이블 내에서 검색하기를 원하는 레코드를 찾기 위한 조건절을 제시한 후, 3) 검색된 레코드 중에서 출력을 통해 살펴보고 싶은 속성들을 선택하는 세 단계의 작업으로 구성된다. VIOLA 역시 이와 유사한 형태의 시각 질의 명세 방법을 제시한다. 다만, SQL 명세 방법과의 차이점은 테이블 대신 클래스를, 속성뿐만 아니라 관련성과 연산도 명세할 수 있으며, 다양한 컬렉션(collection) 타입과 복합 객체(complex object)를

기술할 수 있으며, 질의의 중첩이 가능하다는 점이다. 두번째 특징은 시각 질의 명세에 미숙한 사용자들이 쉽게 익혀 사용하기에 편리하도록 하기 위해 반드시 지원되어야 하는 특징이다. VIOLA는 기본이 되는 시각 구조(visual construct)와 부가적인 몇 가지의 시각 요소(visual element)만을 통해 초보 사용자가 쉽게 질의를 명세할 수 있도록 설계되었다. 또, 숙련된 사용자를 위해서는 보다 복잡한 시각 질의 구조를 명세할 수 있도록 시각 구조의 중첩을 통해 새로운 질의 구조를 유도(derive)할 수 있도록 하는 기능도 제공된다. 세번째 특징은 시각 질의 언어가 실제로 효율적으로 활용될 수 있기 위해서 반드시 만족되어야 할 특징이 된다. VIOLA는 시각 질의 언어의 명세로부터 OQL[2] 질의 문장을 생성하여 준다. 이는 작성된 시각 질의를 OQL 질의로 변환, 저장하여 OQL을 지원하는 어느 객체지향 데이타베이스 상에서도 실행될 수 있도록 하기 위해서이다. 목표 질의어로서 OQL을 선택한 것은 대부분의 상용 객체지향 데이타베이스 회사들이 ODMG(Object Database Management Group)[2]에 참여하고 있으며, ODMG는 OQL을 표준 질의어로 삼고 있기 때문이다. 네번째 특징은 시각 질의 언어가 언어로서의 특성을 갖추기 위해 반드시 만족되어야 할 특징이다. 본 연구에서는 이를 위해 시각 구조의 세만틱을 모노이드 내포 해석식(Monoid Comprehension Calculus)[4]에 기반하여 제시하였다.

본 연구는 객체지향 데이타베이스 관리 시스템인 SOP(SNU OODB Platform)[3][3] 상에서 시각 질의의 작성 및 실행, 객체 검색을 위한 사용자 인터페이스로서 구현되었다. SOP는 ODMG 표준안[4][2]을 지원한다. 따라서, 본 논문에서 설명하는 객체지향 데이타베이스의 기저 데이타 모델은 ODMG 데이타 모델[2]이 된다. 본 논문의 나머지 부분에서는 본 시스템이 어떻게 이런 기능을 구현하였는지와 시스템의 구조, 중요한 특징들에 대해서 설명한다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 본 연구가 기반 데이타 모델로 삼고 있는 ODMG 데이타 모델에 대해서 설명하고, 예제 스키마를 제시한다. 3장에서는 VIOLA의 시각 질의를 구성하는 기본 구조와 시각 요소에 대해서 설명하고, 예제 질의를 통해 시각 질의의 명세 방법을 제시한다. 4장에서는 VIOLA 언어가 가지는 정형적 특징으로서 세만틱과 표현력에 대해서 살펴본다. 5장에서는 VIOLA 언어를 구현한 프로토타입 시스템인 SOPView의 시스템 구조 및 시각 질의의 내부 표현 구조와 OQL 질의로의 변환 방법에 대하여 설명한다. 6장에서는 시각 질의 명세를 지원하여 주는 관련 연구를 통해 기존의 관련 연구 상황에 대해 고찰하여 본다. 그런 후, 7장에서 결론을 맺기로 한다.

[2] ODMG는 객체지향 데이타베이스의 표준 모델을 제정하기 위해 해외의 유명한 상용 객체지향 데이타베이스 시스템 개발 회사들(예를 들면, SunSoft, O_2 Technology, Objectivity, ONTOS, Versant, etc.)이 연합하여 결성한 콘소시엄이다.

[3] SOP는 서울대학교 컴퓨터공학과 객체지향 연구실에서 1993년부터 1995년까지 3년간 개발하여 완성한 객체 지향 데이타베이스 관리 시스템이다.

[4] 현재 이 표준안은 Ver. 1.2까지 나와 있으며, 이미 세계 표준화 기구에 표준안으로 제출되어 있다.

2 데이타 모델

2.1 ODMG 데이타 모델

ODMG 데이타 모델[2]에 있어 가장 기본이 되는 개념은 객체(object)이다. 객체는 실세계를 구성하는 요소들을 나타내며, 상태(state)와 행위(behavior)로 구성된다. 객체의 상태는 성질(property)들의 집합으로, 객체의 행위는 연산(operation)들의 집합으로 정의된다.

객체의 성질은 속성(attribute)과 관련성(relationship)으로 구성된다. 속성은 타입(type)과 리터럴(literal)의 쌍으로 나타내어지며, 객체의 정적인 속성을 나타내게 된다. 관련성은 두 개의 객체들 사이에 정의되어지며, 객체들 사이의 연관 관계를 나타내게 된다. 이러한 두 객체 간에는 항해할 수 있는 여행 경로(traversal path)가 존재하게 된다. 관련성은 'inverse' 키워드의 지정을 통해 역방향 경로(inverse path)가 제공된다. 즉, 한쪽 방향에서 해당 관련성에 대해 변경 및 삭제 등의 연산을 수행하면 이것이 반대편 객체들의 역 관련성에도 반영되어, 참조 무결성(referential integrity)이 보장받을 수 있게 된다. 객체의 연산은 객체가 실행할 수 있는 기능들을 나타낸다.

같은 성질 및 행위를 갖는 객체들은 타입(type)으로 무리지어진다. 타입들 간에는 상속 관계가 존재할 수 있으며, 이 관계들은 계층 구조(hierarchy)를 이룬다.

2.2 예제 스키마

그림 1은 ODMG 모델 상에서의 대학 스키마의 구조를 보이고 있다.

이 스키마에서 'Professor', 'Graduate', 'Student' 및 'Employee' 클래스는 대학의 구성원들을 모델링하고 있다. 'Person'과 'Univ-Person' 클래스는 이들의 수퍼클래스(superclass)이다. 'Department' 클래스는 대학 구성원들이 속한 학과를, 'Course' 클래스는 대학에서 개설된 강의를 모델링한다. 'Enroll' 클래스는 'Student'와 'Course' 클래스 사이의 수강 관계를 모델링한다. 'Person' 클래스의 *colleague* 관련성은 자기 자신을 도메인으로 하는 재귀적(recursive) 관련성이다. 이 관련성은 같은 'Person' 클래스에 속한 다른 객체들의 집합을 참조한다. 이는 한 사람이 다른 여러 사람을 동료로 가질 수 있음을 나타낸다. 또, 'Univ-Person' 클래스의 *dept* 관련성은 'Department' 클래스를 도메인으로 가지는데, 이 관련성은 단지 하나의 객체만을 참조하는 구조이다. 즉, 한 대학 사람은 하나의 과에만 속함을 나타낸다. 이 두 관련성은 다른 속성들과 함께 모두 'Professor', 'Graduate', 'Student' 및 'Employee' 클래스들로 상속된다. 'Professor' 클래스의 객체들은 *supervises* 관련성을 통해 다수의 'Graduate' 클래스의 객체들을 참조하여 교수의 대학원생 지도 관계를 모델링 하고 있으며, *lectures* 관련성을 통해 다수의 'Course' 객체들을 참조하여 교수가 강의하는 과목들을 나타내고 있다. 반대로, 'Graduate' 클래스

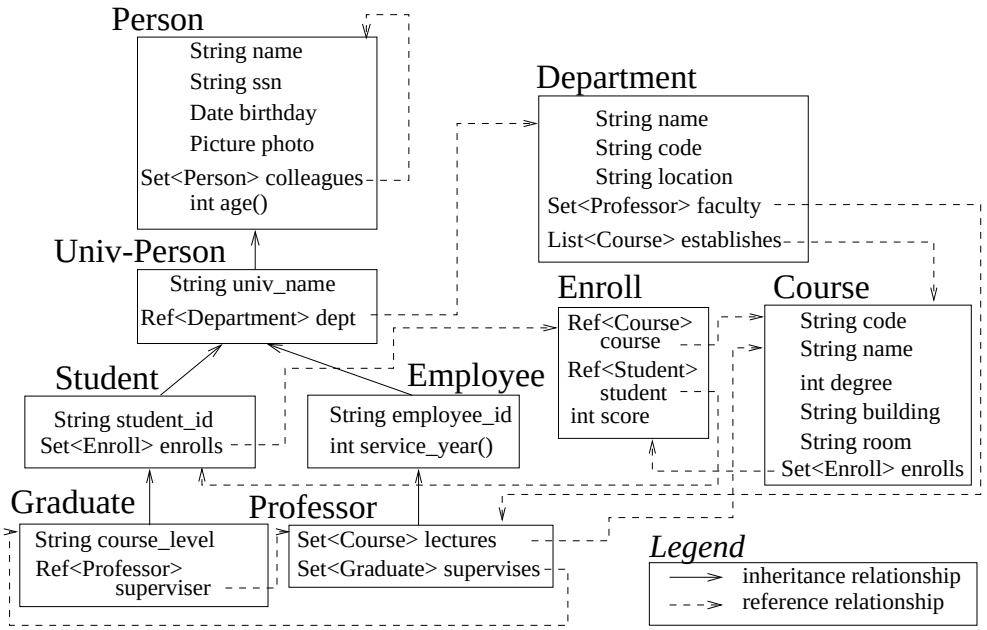


그림 1: ODMG 모델 상에서의 예제 스키마 구조(대학 데이타베이스)

의 객체들은 *superviser* 관련성을 통해 하나의 'Professor' 클래스의 객체를 참조하는데, 대학원생들은 단지 한명의 지도 교수를 가짐을 나타낸다. 'Student' 클래스의 객체들은 *enrolls* 관련성을 통해 다수의 'Enroll' 객체들을 참조하는데, 이는 학생들의 수강 내역을 나타낸다. 'Enroll' 클래스는 다시 *course* 관련성을 통해 'Course' 과목을 참조해서 수강하는 과목을 나타낸다. 'Department' 클래스는 *faculty* 관련성을 통해 다수의 'Professor' 클래스의 객체들을, *establishes* 관련성을 통해 다수의 'Course' 클래스의 객체들을 참조하는데, 이는 각각 학과의 교수진과 개설 과목을 모델링하고 있다. 이 대학 스키마는 뒤에서 시각 질의 예제를 보일 때 기반 스키마로서 사용하기로 한다.

3 시각 질의 언어

본 장에서는 객체지향 데이타모델에 기반한 시각 질의 언어인 VIOLA의 기본 구조와 시각 요소에 대해서 설명한다. 그리고, 예제 질의를 통해 시각 질의의 명세 방법에 대해 살펴 본 후, VIOLA와 OQL과의 관련성에 대해서 제시한다.

3.1 시각 질의 구조

VIOLA에서 시각 질의의 명세를 위한 기본 구조와 이에 대응하는 OQL 문장은 다음의 그림 2와 같다. 이 기본 구조는 크게 머리(head) 부분과 몸체(body) 부분으로 구성된다. 머리 부분은 프로젝션(projection)

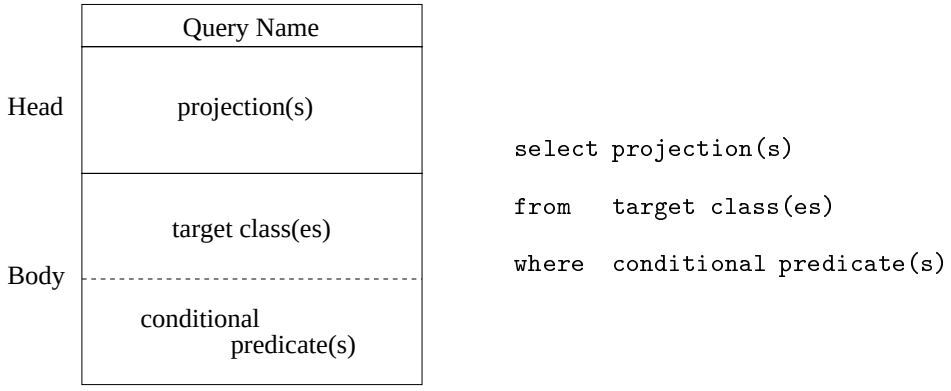


그림 2: 시각 질의 기본 구조와 대응하는 OQL 문장

명세 부분으로 구성된다. 몸체 부분은 다시 목표 클래스(targe class) 명세 부분과 조건절(conditional predicate) 명세 부분으로 나뉘어진다. 먼저, **프로젝션 부분**에는 밑의 몸체 부분의 시각적 명세를 통해 검색되어진 객체들에 대해 사용자가 실제로 결과로써 받아보고 싶은 부분을 명세하게 된다. 이 때에는 객체 전체를 지정할 수도 있고, 일부분을 지정할 수도 있고, 서로 다른 여러 부분들을 모아 새로운 구조체로서 리턴받도록 지정할 수도 있다. 또, 이들 프로젝션 부분들을 저장할 컬렉션의 타입도 지정하게 된다. 다음으로 **목표 클래스 부분**에는 질의를 실행할 목표 클래스를 지정하게 된다. 목표 클래스로는 스키마에 속한 클래스들 뿐만 아니라 다른 질의 자체가 중첩되어 명세될 수도 있다. 마지막으로, **조건절 부분**에는 목표 클래스들에 속하는 객체들 중에서 특정 조건을 만족하는 객체들만을 찾아내기 위한 조건절들이 명세되어진다. 또한, 두 개 이상의 클래스들이 지정되었을 때, 이들 사이의 조인(join) 관계식 또한 명세되어진다. 부가적으로 질의명은 사용자가 특정 질의를 재사용하기 위해 이름을 명명하였을 때, 이 질의명을 나타내게 된다. 이상의 기본 구조와 이것이 가지는 세만틱은 기존의 SQL 문장과 거의 동일하여, 질의 명세시 사용자의 혼란을 줄이고 호환성을 제공하게 된다.

3.2 시각 요소

VIOLA는 몇 가지의 간단한 시각 요소들만을 제공하는데, 시각 질의의 명세를 위한 시각 요소는 다음의 그림 3과 같다. 먼저 네모의 사각형은 객체들의 모임인 컬렉션을 나타낸다. 클래스에 속한 모든 객체들을 나타내는 익스텐트(extent) 및 집합(set) 타입이 이 형태로 표현된다. 반면, 모서리가 둥근 사각형은 단일의 객체를 나타낸다. 여기에는, 리터럴(literal) 및 이름을 가지는 객체(named object)가 표현될 수 있다. 모든 컬렉션에는 대문자의 변수가 지정되는데, 이는 컬렉션에 속한 모든 객체들을 나타낸다. 컬렉션에 속한 하나의 객체를 지정하는 변수는 같은 알파벳의 소문자로 표기한다. 이 변수는 변수가 처음 지정된 시각 구조와 이 안에 중첩된 시각 구조 내에서 유효(valid)하게 된다. ODMG 데이타 모델

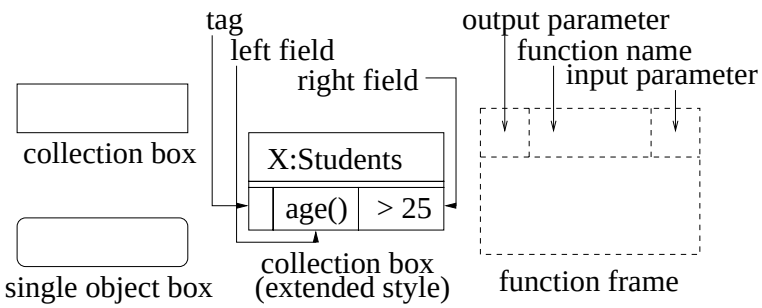


그림 3: 시각 질의 명세를 위한 시각 요소들

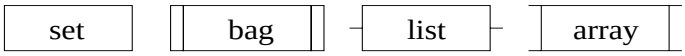


그림 4: 컬렉션의 타입별 표기 방법

은 네 가지 타입의 컬렉션(set,bag,list, 그리고 array)을 지원하는데 컬렉션을 타입별로 구분해서 표기하기 위한 시각 요소는 그림 4와 같다. 컬렉션 및 단일 객체 표현은 조건절 부분 안에서 조건절 및 경로 표현(path expression)을 나타내기 위해 확장될 수 있다. 조건절은 확장된 부분에 속성명 혹은 연산(메소드)과 이에 대한 함수가 축약된 형태로 표현되는데, 이는 컬렉션에 속한 객체들 중에서 해당 조건을 만족하는 객체들만을 선택하기 위한 표현이 된다. 연산 호출에 인자가 필요한 경우에는 괄호를 통해 인자를 명세한다. 또, 경로 표현을 위해서는 함수 대신 변수가 지정되는데, 속성 혹은 연산에 변수가 지정된 경우 이 변수는 시각 질의의 다른 부분에서 해당 속성 혹은 연산을 나타내기 위해 사용되어진다. 관련성에 변수가 지정된 경우에는 관련성이 해당 변수가 지칭하는 컬렉션(카디낼러티가 'm'인 경우) 혹은 단일 객체(카디낼러티가 '1'인 경우)를 참조하고 있음을 나타낸다. 이 관련성 변수는 경로 표현 상의 한 경로를 나타내게 된다. 하나의 컬렉션 혹은 단일 객체로부터 시작된 일련의 변수들이 모여 하나의 경로 표현을 나타내게 된다.

컬렉션 및 단일 객체에 대한 연산들은 조건절 부분에서 함수 프레임(function frame)을 통해 명세될 수 있다. 이 함수 프레임은 함수의 이름 및 함수에 대한 입력 인자와 실행 결과인 출력 인자를 명세할 수 있도록 되어 있다. 입력 인자로 명세된 변수들은 함수 프레임 이외의 곳에서 컬렉션, 단일 객체 및 속성, 관련성 등에 지정된 변수들로서, 이들 변수들을 통해 함수에 대한 입력 요소를 지정하게 된다. 출력 인자로 명세된 변수는 이 함수의 결과값 혹은 결과 객체를 나타내게 된다. 출력 변수 역시 함수 프레임이 정의된 시각 구조와 이 안에 중첩된 시각 구조 내에서 유효하게 된다. 함수들 사이의 조합은 하나의 함수 프레임 안에 다른 함수 프레임을 시각적으로 중첩시키거나, 또는, 하나의 함수 프레임의 출력 변수가 다른 함수 프레임의 입력 인자로 주어지도록 해서 표현하게 된다. 함수 프레임의 결과가 다른 곳에서 사

용되지 않는 경우에는 출력 인자를 명세하지 않아도 된다. 함수 프레임을 통한 다양한 함수의 표현 방법은 그림 5와 같다.

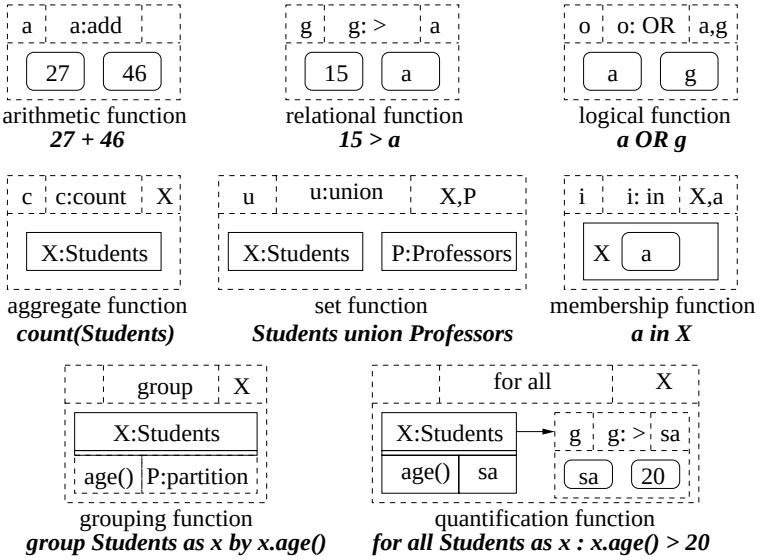


그림 5: 함수 프레임을 이용한 다양한 연산자의 표현 방법

함수 프레임 표현 중에서 객체의 속성 및 연산과 함께 사용되는 산술(arithmetic) 함수와 관계(relational) 함수는 앞의 그림 3의 확장 형태와 같이 축약되어 표현될 수 있다. OQL은 그룹화 함수에 대해 특별한 세만틱을 제공하는데 그룹지어진 객체의 집합을 하나의 컬렉션인 'partition'으로 표현하여 이를 목표로 하여 다시 질의를 작성할 수 있다는 점이다. 이러한 세만틱을 지원하기 위해, 그룹화 함수 프레임 안에서는 그룹화 조건과 'partition' 컬렉션이 함께 표시되도록 한다. 이 표시를 위한 컬렉션 확장에는 조건절 확장과 구분하기 위해 실선이 아니라 점선을 사용한다. 또, 한정자 함수의 표현은 방향선으로 연결된 두 부분으로 표현한다. 방향선의 왼쪽에는 한정 연산의 대상이 되는 컬렉션을, 오른쪽에는 조건절을 나타내는 함수 프레임을 명세한다. 또한, 시각적으로 혼란을 유발할 수 있는 복잡한 식은 그림 6과 같이 함수 프레임의 특별한 형태인 조건절 프레임을 통해 곧바로 입력될 수 있다.

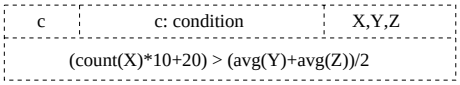


그림 6: 조건절 프레임을 이용한 연산자 표현

3.3 시각 질의 명세 방법

VIOLA는 'Drag & Drop'과 팝업(pop-up) 메뉴에 기반한 시각 질의 명세 방법을 제공한다. 컬렉션과 단일 객체 시각 요소는 모두 윈도우 상의 버튼으로 표시되며, 이들 요소들은 스키마 계층 구조를 나타내는 윈도우 상에서 명세하고자 하는 클래스를 마우스로 끌어 당겨(drag) 시각 구조 상의 원하는 부분에 갖다 놓으면(drop) 된다. 이들 요소의 확장을 위해서는 해당 버튼을 누르면 된다. 그러면, 시각 요소가 나타내는 클래스가 가지는 속성 및 관련성이 팝업(pop-up) 메뉴 상에 나타나게 된다. 이 때, 조건절을 명세하기 위한 속성 혹은 관련성 버튼을 누르면 조건절 및 변수 입력을 위한 대화 상자가 나타나게 되며, 이 안에 원하는 명세 내용을 입력하면 된다. 관련성의 경우, 버튼을 더블 클릭하면 관련성이 참조하는 클래스가 자동적으로 출력된다. 이 클래스에 대해서도 같은 방법으로 시각 명세를 실행할 수 있게 된다. 이 때, 관련성의 카디낼러티(cardinality)가 '다수'인 경우에는 참조되는 컬렉션이 자동적으로 목표 클래스로서 지정된다. 이는 참조 경로를 통한 항해(traversal)시 다수의 관련성은 질의 처리시 모호함(ambiguity)을 주기 때문이다. 함수 프레임은 함수들의 목록에서 원하는 함수를 'drag & drop'을 통해 시각 구조 상으로 가져오면 된다. 그러면, 함수 프레임을 나타내는 조그만 윈도우가 나타나며, 이 안에 함수 프레임 구성 요소들을 앞서 설명한 방법으로 명세하면 된다. 시각 질의의 명세 예제가 그림 7에 나와 있다.

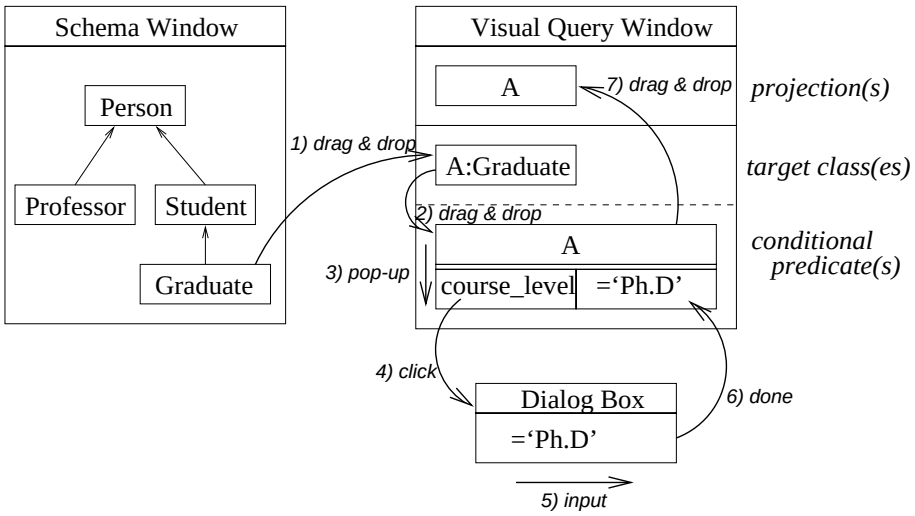


그림 7: VIOLA 시각 질의의 명세 방법의 예제

3.4 예제 질의

이 절에서는 시각 구조 및 시각 요소를 이용한 시각 질의의 작성 예제를 보이기로 한다.

예제 질의 1 : "컴퓨터공학과에 속한 학생들 중에서 'C++' 수업을 듣는 학생들을 찾아라".

이 질의를 위해 사용자는 먼저 '학생' 클래스를 목표 클래스 부분에 'drag & drop'으로 명세한다. 그리고, '학생들' 컬렉션의 버튼을 눌러 확장한다. 먼저, 첫번째 조건절의 명세를 위해 다음의 명세를 수행한다; 1) 팝업 메뉴 상에서 '등록(enrolls)' 관련성을 더블 클릭, 2) 나타난 '등록' 컬렉션(변수명 'B')을 확장한 후 다시 '과목(course)' 관련성을 더블 클릭, 3) 나타난 '과목' 객체(변수명 'cs')를 확장하여 '이름' 속성값을 'C++'로 입력한다. 다음으로, 두번째 조건절의 명세를 위해 다음의 명세를 수행한다; 1) '학생들' 컬렉션의 팝업 메뉴에서 '학과(dept)' 관련성을 더블 클릭, 2) 나타난 '학과(Department)' 객체(변수명 'dp')의 '이름' 속성값을 '컴퓨터공학'으로 지정한다. 이제, '학생들' 컬렉션(변수명 'A')을 프로젝션 부분으로 'drag & drop'으로 명세한다. 이 질의를 나타내는 시각 구조는 그림 8과 같다.

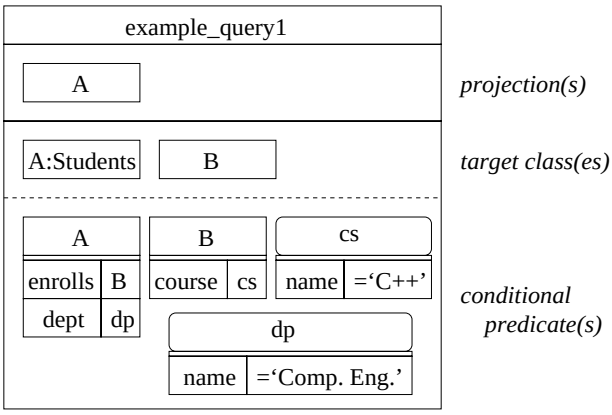


그림 8: 예제 질의 1의 시각 명세

또, 그림 8의 시각적 명세로부터 생성되어지는 OQL 질의 문장은 다음과 같다 :

```
select a
from Students as a, a.enrolls as b
where b.name = 'C++' and a.dept.name = 'Comp. Eng.'
```

예제 질의 2 : "컴퓨터공학과에서 개설한 데이타베이스 과목을 듣는 학생들을 학과별로 분류하여 성적 평균이 80점 이상인 학과명과 성적 평균을 보여라".

이 질의를 명세하기 위해서는 먼저 1) 컴퓨터공학과에서 개설한 데이타베이스 과목에 대한 수강 내역을 찾고, 그런 후, 2) 1)번 질의의 결과인 수강 내역에 대해 해당 수강 내역을 신청한 학생들을 찾아 학과별로 그룹을 나눈 후, 그룹별로 성적 평균을 구해 평균이 80점 이상인 그룹의 학생들이 속한 학과명과 성적 평균을 출력한다. 지면 관계상 이 예제를 위한 구체적인 명세 방법은 [6]을 참고하기 바란다. 이 시각적 명세를 나타내는 질의 윈도우는 그림 9와 같다.

또, 그림 9의 시각적 명세로부터 생성되어지는 OQL 질의 문장은 다음과 같다 :

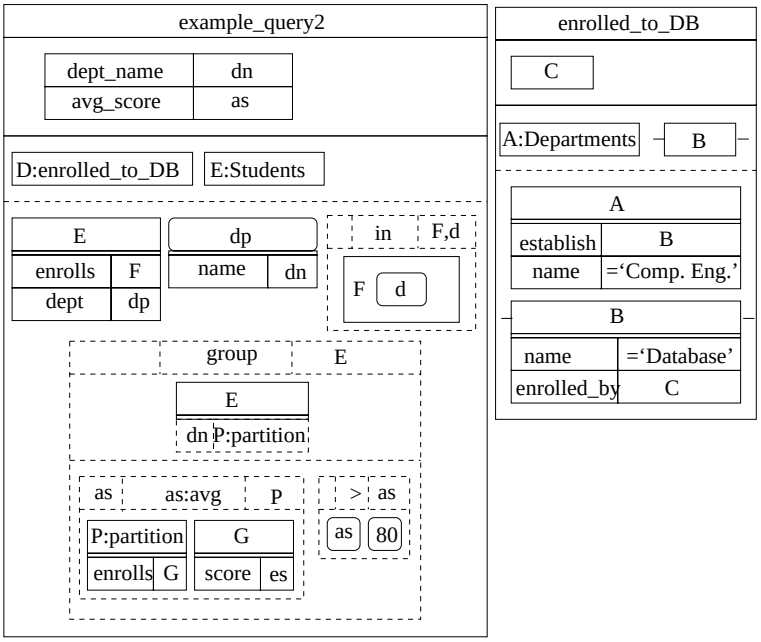


그림 9: 예제 질의 2의 시각 명세

```
select dept_name: e.dept.name,
       avg_score: avg(select g.score from partition as p, p.enrolls as g),
from (select b.enrolls from Departments as a, a.establishes as b
      where a.name = 'Comp. Eng.' and b.name= 'Database') as d,
     Graduates as e
where d in e.enrolls
group e by e.dept.name
having avg(select g.score from partition as p, p.enrolls as g) > 80
```

3.5 VIOLA와 OQL과의 관계

VIOLA에 의해 표현될 수 있는 시각 질의는 OQL로 표현될 수 있다. 하지만, VIOLA가 모든 OQL 질의를 표현할 수 있는 것은 아니다. 이는 앞서 설명한 대로 VIOLA가 기존 범용 질의어와의 호환성 및 이해의 편리성을 위해 'selection & projection' 원칙에 기반한 시각 구조를 기본 구조로 삼고 있기 때문이다. 따라서, 모든 VIOLA 질의는 OQL의 'select from where' 문장으로 바뀌어지게 된다. 하지만, OQL은 프로그래밍 언어와 비슷한 형태의 질의도 지원한다. 예를 들어, '1+2'는 OQL 상에서의 유효한 질의(결과로서 정수 리터럴 3을 리턴하는)가 된다. VIOLA에서는 이 문장을 단독으로는 표현하지는 못한다. 하지만, 이는 VIOLA의 표현력이 OQL에 비해 현저히 떨어짐을 의미하는 것은 아니다. 질의의 중첩 구조 상에서 가장 상위의 표현이 *select* 문장으로 제한될 뿐이지, 기타 표현들이 조건절로 제시될 때, 앞서의 시각 요소들을 통해 표현할 수 있게 된다. 예를 들어, 다음의 OQL 질의 'count(Students)>10'는 직접

표현할 수 없으나, 똑같은 문장이 조건절로 명시된 'select from where' 문장, 즉, 'select x from Students x where count(x.friends)>10'은 VIOLA에 의해 표현될 수 있다.

4 정형적 성질

4.1 시각 질의의 세만틱

이 절에서는 앞에서 설명한 VIOLA의 시각 구조가 가지는 세만틱에 대해서 살펴 보기로 한다. 앞서 살펴본 대로 시각 질의를 작성하는 그래픽 연산은 크게 다음과 같은 세가지, 즉, 1) 목표 클래스 명세 연산, 2) 조건절 명세 연산 및 3) 프로젝션 명세 연산으로 구분된다. 이제 이 세가지의 연산이 가지는 세만틱을 모노이드 내포(monoid comprehension)를 이용하여 기술하기로 한다.

모노이드 내포는 동시에 하나 이상의 컬렉션 타입을 처리 및 조합해야 하는 질의를 표현하기 위한 일관적인 방법을 제공하여, 함수 표현의 통합을 쉽게 수행할 수 있는 장점을 제공한다[4]. 모노이드 내포의 구문(syntax)은 $M\{e|c_1, c_2, ..., c_n\}$이다. 이것은 '$c_1$과 c_2 ...와 c_n을 만족하는 모든 e의 집합을 찾아낸 후, 이것을 모노이드 M이 생성하는 타입으로 변환하라'는 의미가 된다. 모노이드(monoid)는 데이타 타입을 위한 일반적인 템플릿이다[4]. 이것은 관계형 데이타베이스 뿐만 아니라 객체지향 데이타베이스에서 사용되고 있는 대부분의 컬렉션을 표현할 수 있다. 내포(comprehension)[5]는 컬렉션을 지정하는 표현이다[5]. 보통 객체지향 데이타베이스는 set, bag, list, array 등의 많은 컬렉션 타입을 지원한다. 이러한 컬렉션 타입들은 내포를 통해 일관되게 표현되고 처리되어질 수 있게 된다. 위에서, 문장 e는 내포의 머리(head)라고 불리운다. 이것은 집합에 포함되는 요소들을 나타내는 표현이 된다. 또, c_i는 다음의 두 가지 형태를 가진다 :

- $e \leftarrow u$: 이것은 생성자(generator)라 불리우며, 컬렉션 u에 속한 모든 원소를 영역으로 하는 변수 e를 지정하는 것이다.
- *pred* : 이것은 조건을 지정하는 절(predicate)을 나타낸다. 여기에는, v *in* s, $v = v'$, $v \leq v'$등과 같은 간단한 조건들뿐만 아니라, 논리곱(conjunction) *C and C'*, 논리합(disjunction) *C or C'*, 부정(negation) *not C*, 한정자(quantifier) 등이 올 수 있다.

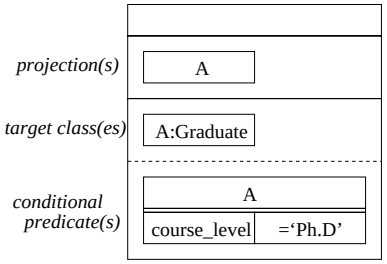
이제, 예를 들어, 다음의 모노이드 내포를 살펴 보자 :

$$set\{e|e \leftarrow Graduate, e.course\_level = `Ph.D'\} - (1)$$

[5] 집합론에서 집합을 표현하는 방법으로는 집합의 원소를 일일이 나열하는 외포적 방법과 집합의 원소들이 공통적으로 가지는 특징을 기술하는 내포적 방법의 두 가지가 있다.

이것은 대학원생(Graduate) 컬렉션에 속한 모든 원소들에 대해서 이 원소가 가지는 '학위과정' 속성값이 '박사과정(Ph.D)'인 모든 원소들의 집합을 찾아 이를 컬렉션 모노이드 'set'이 나타내는 타입으로 변환하라는 의미가 된다.

이제 이러한 정형적인 성질과 시각 구조와의 관계는 다음과 같다; 1) 프로젝션 명세 연산은 모노이드 내포의 구문 중에서 모노이드 및 내포의 머리 부분에 해당하며, 2) 목표 클래스 명세 연산은 내포의 구문 중에서 생성자에 해당하며, 3) 조건절 명세 연산은 위의 내포의 구문 중에서 조건절 지정에 해당한다. 앞의 (1)의 모노이드 내포를 나타내는 시각 구조와 OQL 질의 문장은 다음의 그림 10과 같다. 여기




```
select a
from Graduates as a
where a.course_level = 'Ph.D'
```

그림 10: (1)의 모노이드 내포의 시각 명세와 OQL 질의 문장

서, 생성자 부분은 목표 클래스 명세 부분에 표시되었다. 이는 대학원생 클래스가 질의의 목표 클래스가 되며, 변수 e는 이 클래스의 객체들을 표시함을 나타낸다. 조건절 명세 부분에는 학위 과정이 박사 과정임을 나타내는 조건절이 표시되었다. 그리고, 프로젝션 부분에 대학원생 클래스의 객체들을 표현하여, 조건절을 만족하는 객체들이 결과로서 리턴됨을 나타내었다. 또한, 조건절에 나타난 함수 프레임은 함수의 종류에 따라 동등한 다른 모노이드 내포 문장으로 변환된다. OQL 함수와 모노이드 내포 사이의 관계는 [4]에 잘 나와 있다.

4.2 표현력

VIOLA의 부분집합으로 관계형 대수(relational algebra)[7] 질의를 충분히 표현할 수 있다. 관계형 대수의 중요 연산으로는 *selection, projection, join, union, difference* 등이 있다. 선택 연산은 조건절의 제시에 의해 수행된다. 프로젝션 연산은 프로젝션 부분에 대한 시각적인 명세로서 수행된다. 조인 연산은 목표 클래스에 대한 명세로서 수행된다. 합집합 연산과 차집합 연산은 함수 프레임을 통해 수행된다. 이제 이상의 시각 요소로 구성된 시각 질의를 R(elational)-VIOLA라고 하자. 그러면, R-VIOLA는 관계형 대수 연산을 모두 제공하므로 관계형 대수와 동등하고, 따라서, 관계적으로 완전(relationally complete)하다. 또, 관계형 대수는 관계형 해석식(relational calculus)와 동등하고[8], 관계형 해석식은

집합을 포함하는 관계형 해석식과 동등하므로[9] R-VIOLA는 관계형 대수, 관계형 해석식, 집합을 포함하는 관계형 해석식과 동등하게 된다. 여기서, R-VIOLA는 VIOLA의 부분 집합이고, SQL이 관계형 대수 연산 이외에 포함하는 연산들(한정자, 집단화 및 그룹화 연산)은 VIOLA도 역시 지원하므로, VIOLA의 모든 시각 요소에 의한 표현력은 객체 검색에 관한한 적어도 SQL[10]이 가지는 표현력과 같거나 크게 된다. VIOLA와 SQL 및 OQL과의 비교는 다음의 표 1과 같다.

	SQL	*VIOLA*	*OQL*
Relational algebra operations (selection,projection,join,union,difference)	O	O	O
Quantifications(universal,existential)	O	O	O
Aggregate functions(sum,avg,count,min,max)	O	O	O
Grouping & Ordering	O	O	O
Collection support	X	O	O
Path expression	X	O	O
Method support	X	O	O
Composition of every function	X	△	O

O : 지원, X : 지원 않음, △ : 부분적으로 지원

표 1 : VIOLA와 SQL 및 OQL과의 비교표

5 프로토타입 시스템 : SOPView

본 장에서는 VIOLA 시각 질의 언어의 기능 중에서 한 부분을 구현한 프로토타입 시스템인 SOPView에 대하여 설명한다. SOPView 시스템은 객체지향 데이타베이스 관리 시스템인 SOP 상에서의 시각 질의 및 객체 검색을 실행할 수 있도록 도와 주는 그래픽 사용자 인터페이스 프로그램이다[19]. SOPView에는 VIOLA의 특징 중 기본 시각 구조, 기본 시각 요소 및 함수 프레임의 일부분과 조건절 박스 등이 구현되어 있다. 먼저 시스템의 윈도우 구조 및 내부 모듈 구성에 대해서 설명한 후, 다음으로, 시각 질의의 표현을 위한 내부 자료 구조에 대해 제시한다. 그런 후, 이 내부 구조를 OQL 언어로 변환하는 방법에 대해서 보이기로 한다. 시각 질의의 명세 방법 및 데모 실행의 예는 [19]에 잘 나와 있다.

5.1 시스템 구성

SOPView 시스템의 화면의 구성은 다음의 그림 11과 같다. 이것은 그림 7에 나타난 시각 질의를 보이고 있다. SOPView 시스템은 **스키마 윈도우**(schema window), **질의 객체 윈도우**(query object window), 그리고, **질의 윈도우**(query window)로 구성되어 있다. 사용자는 먼저 스키마 윈도우를 통해 스키마 구조를 살펴보게 되는데, 스키마 윈도우 상에는 스키마 구조가 DAG(Directed Acyclic Graph)의 형태로 출력된다. 그런 다음, 사용자는 질의 윈도우에 시각 질의를 명세하게 된다. 질의 윈도우는 크게 목

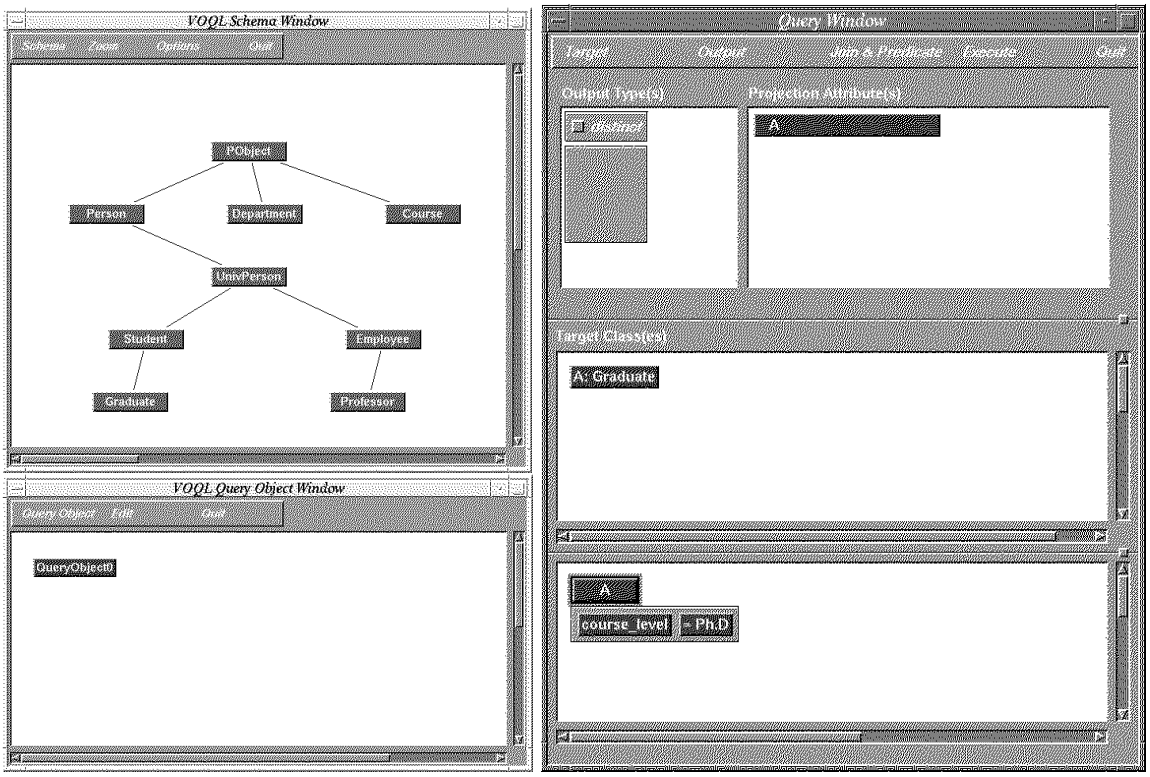


그림 11: SOPView의 화면의 구성

표 클래스 창(target class pane), 조건 창(condition pane), 프로젝션 창(projection pane), 출력 타입 창(output type pane) 등으로 구성된다. 이 질의 윈도우에 대한 명세 방법은 앞서 살펴본 바와 같다. 하나의 질의 윈도우는 질의 객체 윈도우 상에 하나의 질의 객체로 표현되어진다. 질의들은 중첩될 수 있는데, 여러 질의 객체들간의 중첩 구조는 질의 객체 윈도우를 통해 표현된다. 질의의 명세를 끝마친 후 질의를 수행하면, 이 질의 명세는 OQL 질의 문장으로 변환되어 SOP의 질의 처리기에 의해 실행되어지게 된다.

본 시스템을 구현하기 위해 기반 환경으로는 Unix & X 윈도우 시스템을, 프로그래밍 언어로는 GNU C++[11]를, 그래픽 사용자 인터페이스 라이브러리로는 Motif[12]를 사용하였다. 시스템은 현재 약 22,000 라인 정도의 C++ 코드로 구성되어 있다.

5.2 시각 질의 표현 구조

이 절에서는 앞 절에서 설명한 시각 질의 명세 도구에서 시각 질의를 내부적으로 표현하기 위해 취하고 있는 클래스 계층 구조 및 시각 요소 객체들의 구조에 대해서 살펴 본다. 먼저 시각 질의 구성 요소들은 객체지향적으로 모델링되었다. 이들을 모델링한 클래스 구조는 다음의 그림 12와 같다. 이 모델링 구조

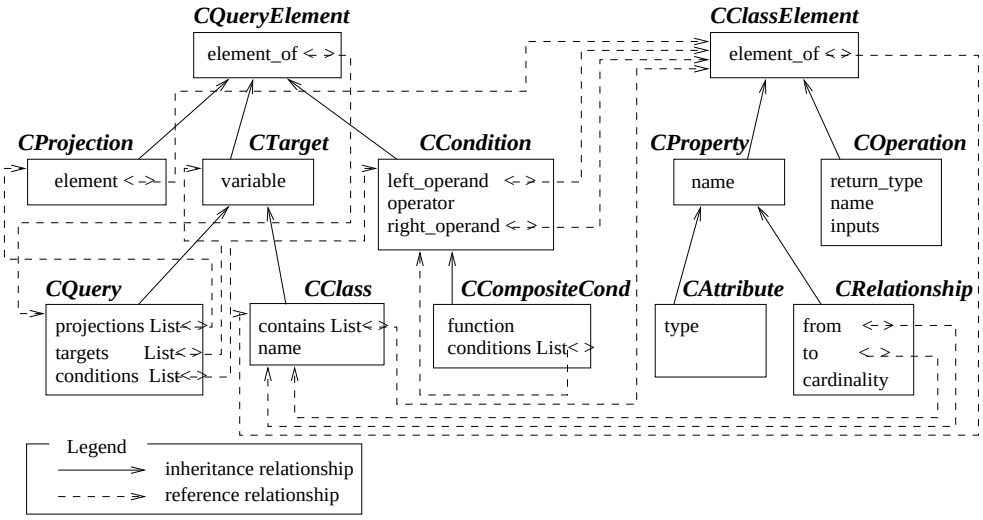


그림 12: 시각 질의 구성 요소들의 모델링 구조

는 크게 두 개의 클래스 계층 구조로 구성된다. 하나는 시각 질의를 구성하는 요소들을 모델링하고 있는데, *CQueryElement* 클래스를 루트 클래스로 하며, 프로젝션에 지정된 요소들을 나타내는 *CProjection* 클래스, 목표 클래스로 지정된 요소들을 나타내는 *CTarget* 클래스, 조건절로 지정된 요소들을 나타내는 *CCondition* 클래스를 서브 클래스로 갖는다. 또, *CTarget* 클래스는 *CClass* 클래스와 *CQuery* 클래스를 서브 클래스로 갖는데, 이는 목표 클래스로서 스키마에 속한 클래스 뿐만 아니라 다른 질의문도 지정될 수 있음을 나타낸다. 여기서, 중요한 것은 *CQuery* 클래스가 *CProjection*, *CTarget* 및 *CCondition* 클래스를 리스트 구조를 통해 참조하고 있다는 것이다. 이는 하나의 질의가 프로젝션들, 목표 클래스들 및 조건절들의 리스트로 구성됨을 나타낸다. 다른 하나의 계층 구조는 ODMG 데이타 모델에서 정의하고 있는 클래스 구조(속성, 관련성 및 연산)를 구성하는 요소들을 모델링하고 있다.

하나의 시각 질의는 그림 12에 나타난 클래스들의 객체들로 구성된다. 앞서 살펴본 예제 1의 시각 질의를 구성하는 시각 요소 객체들의 내부 참조 구조는 다음의 그림 13과 같다. 여기서, 객체들 사이의 참조 구조의 가장 상위에는 하나의 질의 객체가 존재해서 하나의 시각 질의를 대표하게 된다. 이 질의 객체는 각각 프로젝션 객체들, 목표 클래스 객체들 및 조건절 객체들 등의 질의 요소들의 리스트를 유지한다. 이들 질의 요소 객체들은 해당 요소에 지정된 실제 클래스 멤버들, 즉, 클래스의 속성, 관련성 혹은 연산을 나타내는 객체들을 가리키도록 구성된다.

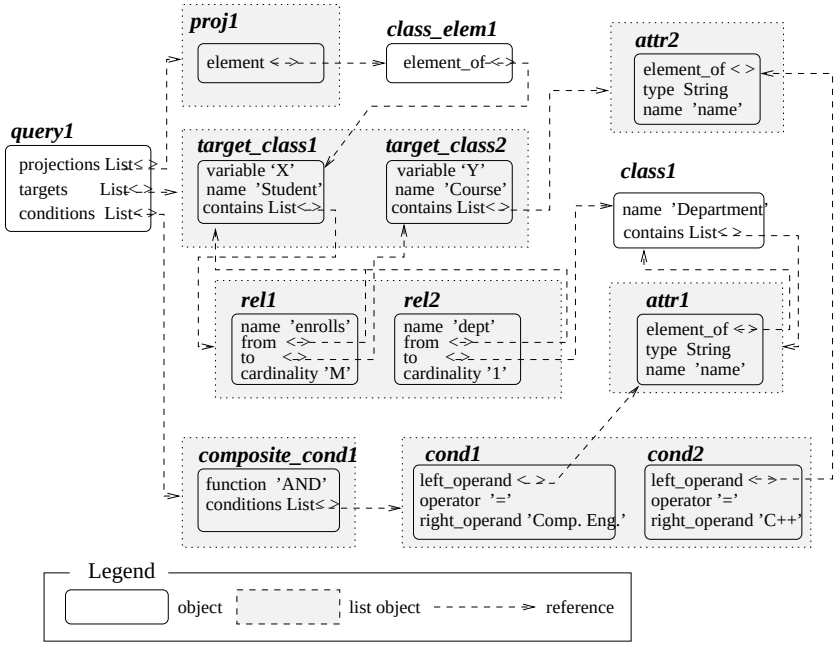


그림 13: 예제 1의 시각 질의를 구성하는 시각 요소들의 객체 구조

6 관련 연구

현재 다양한 시각 질의 언어들이 나와 있는데, 이들 관련 연구를 잘 정리해 놓은 작업으로 C. Batini *et al.*[13]과 G. Ozsoyoglu *et al.*[14]을 들 수 있다. C. Batini *et al.*은 시각 질의의 기본 구성 요소 및 시각 질의의 작성을 지원하는 다양한 접근 방법들에 대해서 설명하고 있다. G. Ozsoyoglu *et al.*은 예제에 기반한 시각 질의 작성 기능을 지원하여 주는 다양한 시각 질의 언어들에 대한 특징별 분류 및 기본 개념에 대하여 제시하고 있다. 본 연구에 큰 영향을 준 시각 질의 언어는 VQL[15]이다. 이 언어는 제한된 광역 한정자(restricted universal quantifier)를 사용하여 관계형 모델 및 중첩 모델, 그리고, 객체 지향 모델에 대해서도 일관되게 시각 질의를 수행할 수 있음을 보이고 있다. VQL은 중첩 윈도우에 기반한 시각 구조를 갖고 있는데, 이 시각 구조에 대한 고찰이 본 연구에 많은 도움을 주었다. 하지만, VQL의 시각 구조가 Datalog에 기반한 세만틱을 가지는데 비해, VIOLA의 시각 구조는 모노이드 내포에 기반한 세만틱을 가지는 차이가 있다. GRAQULA[16]는 ER 모델 혹은 관계형 모델 상에서의 시각적 명세를 지원하는 시각 질의 언어이다. 특히, 이 언어는 이차원적으로 한정 연산을 실행하는 세밀한 방법을 제시하고 있다. 이를 위해 다양한 형태의 프레임을 제공하고 있는데, 이에 대한 고찰이 또한 본 연구에 영향을 주었다. 하지만, GRAQULA의 프레임이 논리 연산만을 표현하는 반면, VIOLA의 함수 프레임은 집합 연산 및 집단화 연산도 포함한다. 또, VIOLA의 함수 프레임은 함수 구조에 기반하고 있다. 본 연구와의 차이점은 GRAQULA는 ER 모델 및 관계형 모델을 지원하는 반면 VIOLA는 객체지향 모델

을 지원한다는 점이다. 이 밖에도 여러 시각 질의 언어들이 있는데, QBD*[17]는 ER 모델 상에서 다이어그램에 기반하여 시각 질의를 작성할 수 있도록 한다. 작성된 시각 질의는 먼저 내부 언어로 바뀌며, 이를 다시 목표 DBMS에 맞는 질의어로 바꾸도록 구성되어 있다. QBD*는 ER 스키마 상에서의 항해 연산을 내부 언어로 모델링 하였다. GraphLog[18]는 그래프를 기반으로 한 데이타 모델 상에서 시각 질의 및 결과의 시각화를 지원한다. GraphLog는 그래프 표현 모델을 이용하여 재귀적(recursive) 질의를 쉽게 작성할 수 있도록 도와 주는 장점을 제공한다.

7 결론

본 연구에서는 객체지향 데이타베이스 상에서 시각적으로 질의를 명세할 수 있도록 도와 주며, 작성된 시각 질의를 OQL 질의 문장으로 변환하여 주는 시각 질의 언어인 VIOLA에 대하여 제시하였다. VIOLA 시각 질의 언어는 다음과 같은 특징들을 제공한다 :

1. 사용자는 기존의 범용 질의 언어인 SQL의 질의 명세 방법과 유사한 간단한 원칙만을 따라서 시각 질의를 작성할 수 있다.

2. 초보자는 몇 가지의 간단한 시각 요소만을 가지고 쉽게 시각 질의를 작성할 수 있으며, 숙련된 사용자는 질의 중첩 기능을 통해 복잡한 질의를 명세할 수 있다.

3. 최근 많이 사용되고 있는 질의 언어인 OQL로 변환되어 OQL 질의 문장이 생성된다.

4. 모노이드 내포에 기반한 정형적인 세만틱을 가진다.

향후의 연구로서는 현재 작성된 SOPView 시스템을 인터넷 환경에서의 시각 질의의 명세 및 객체 검색 시스템으로 확장하는 작업을 수행할 계획을 가지고 있다.

참고문헌

[1] W. Kim, "Introduction to Object-Oriented Database", The MIT Press, 1990

[2] R.G.G. Cattel, "The Object Database Standard: ODMG-93, Release 1.1", Morgan Kaufman Publishers, San Mateo, California, 1994

[3] 안정호, 김형주, "SRP에서 SOP까지", 한국 정보과학회 *Review*지, 1994년 4월

[4] L. Fegaras and D. Maier, "Towards an Effective Calculus for Object Query Language", *Proc. of Int'l Conf. on Management of Data,* San Jose, California, 1995

[5] P. Buneman, L. Libkin, D. Suciu, V. Tannen, and L. Wong, "Comprehension Syntax", *SIGMOD RECORD* Vol.23, No.1, pp.87-96, March 1994

[6] 장성우, 김형주, "VIOLA : 객체지향 데이타베이스를 위한 시각 질의 언어", 기술 문서, *http://wwwoopsla.snu.ac.kr/researchers/swchang/TR-VIOLA.ps*

[7] E. F. Codd, "A Relational Model for Large Shared Data Banks", *Communications of the ACM,* Vol. 13, No. 6, pp.377-387, June 1970

[8] E. F. Codd, "Relational Completeness of Database Sublanguages" in Database Systems, *Prentice Hall,* 1972

[9] G. Ozsoyoglu and H. Wang, "A Relational Calculus with Set Operators, Its Safety and Equivalent Graphical Languages", *IEEE Trans. on Soft. Eng.,* Vol.15, No.9, pp.1038-1052, 1989

[10] C.J. Date, "An Introduction to Database Systems, vol.1", 5th ed., *Addison-Wesley,* 1990

[11] B. Stroustrup, "The C++ Programming Language", 2nd Ed., *Addison Wesley,* 1992

[12] Open Software Foundation, "OSF/Motif Programmer's Guide", *Prentice Hall,* 1993

[13] C. Batini *et al.*, "Visual Query Systems", *Technical Report N. 04.91,* Dipartimento di Informatica e Sistemistica, Universita di Roma "La Sapienza", 1991

[14] G. Ozsoyoglu *et al.*, "Example-Based Graphical Database Query Languages", *IEEE Computer,* pp.25-38, May 1993

[15] K. Vadaparty, Y.A. Aslandogan, and G. Ozsoyoglu, "Towards a Unified Visual Database Access", *Proc. of Int'l Conf. on Management of Data,* Washington, DC, 1993

[16] G. H. Sockut, L. M. Burns, A. Malhotra, and K.-Y. Whang, "GRAQULA : A Graphical Query Language for Entity-Relationship or Relational Databases", *Data & Knowledge Engineering,* Vol.11, pp.171-202, 1993

[17] M. Angelaccio, T. Catarci, and G. Santucci, "QBD*: A Graphical Query Language with Recursion", *IEEE Trans. on Soft. Eng.,* Vol.16, No.10, pp.1150-1163, October 1990

[18] M. P. Consens and A. O. Mendelzon, "GraphLog: a Visual Formalism for Real Life Recursion", *Proc. of Int'l Conf. on Management of Data,* San Diego, California, 1990

[19] S.-W. Chang, S. Lee, and H.-J. Kim, "SOPView: A Visual Query and Object Browsing Environment for SOP OODBMS", *Proc. of 20th Int'l Computer Software and Application Conf.,* pp.354-360, 1996